下载APP

10 | 模型优化（下）：交叉验证，同时寻找最优的参数

2021-09-20 黄佳

《零基础实战机器学习》 课程介绍 >

讲述：黄佳
时长 20:09 大小 18.46M

你好，我是黄佳。

欢迎来到零基础实战机器学习。在前面几节课中，我们已经学习了两种优化机器学习模型的方法，一种是做各种特征工程，让特征集更适合模型；另一种是防止过拟合，让模型不用“那么”精确。

这两种方式的优化角度不同，特征工程是从数据预处理的角度，给出更高质量的数据，而防止过拟合则是在模型训练的过程中，控制模型的复杂度。其实，除此之外，我们还可以从模型的验证和评估环节入手，来优化模型性能。

交叉验证：小数据集的资源复用

你知道，在样本充足的情况下，我们会随机将数据分为 3 个部分：训练集、验证集和测试集。其中，训练集用来训练模型，验证集用来模型调优，测试集用来评估模型性能。

不过，你还记不记得我们在第 1 讲中介绍监督学习的时候，我说过有标签的数据是非常难以获得的。本来就很有限的数据集还要被拆成训练集、测试集和验证集，真是让人特别舍不得。

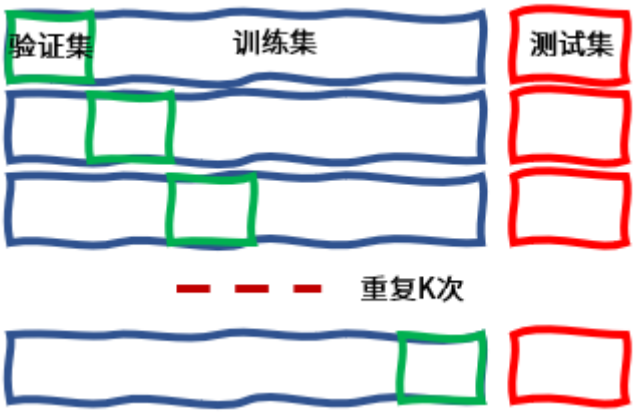
而且，我们知道数据集越大，就越不容易出现过拟合的现象。那么，我们如何利用较小的数据集，从而达到较大数据集的效果呢？这就需要交叉验证。

交叉验证的基本思想是这样的：将训练数据集分为 k 等份，其中 k-1 份用作训练集，单独的那一份用作验证集，整个过程重复 k 次，这也通常称作 k 折。这样就最大程度重复地使用了训练集中的数据，每一个数据都既做了训练，又做了测试，从而在最大程度上提高模型性能的可信度。

这种交叉验证具体实现起来有 4 个步骤：

1. 随机清洗数据集，将数据集分为训练集和测试集，将测试集预留出来，放在一边。
2. 将训练数据集分成 k 组（这个 k 值一般是随机设定，比如 3，5，10，实操中以 10 折居多）。在这一步中，我们挑其中 1 组作为验证集，剩下的 k-1 组做训练集（这个过程要重复 k 次）。我们在这些训练集上拟合模型，可以得到 k 个不同的模型。然后再在对应的验证集上对这些模型进行评估，就能得到一系列模型的评估分数。最后，我们把这些评估分数进行平均，这个平均分数就是交叉验证的最终结果了。
3. 按照步骤 1、2，对多个算法进行交叉验证，比如可以针对线性回归、决策树、随机森林等算法。根据每个算法交叉验证的评估结果，从中挑选效果最好的算法模型。
4. 使用测试集评估最终模型的分数。

现在，你应该很清楚了：每个数据样本都有 1 次机会进入验证集中，并用于训练模型 k-1 次。这样一来，我们就拥有了更多的数据量。整个过程如下所示：



K折交叉验证示意图

在交叉验证中，训练集和验证集的拆分可以通过 sklearn.model_selection 中的 KFold 函数实现。在这个函数中，有三个主要参数需要我们了解一下：

KFold函数主要参数	说明
n_splits	int型，默认值=5，折叠次数，必须至少为 2。
shuffle	bool型，默认值=False，是否在拆分前打乱数据。注意一旦拆分了，每个拆分中的样本并不会被打乱。
random_state	int型，默认值=None，是否设定随机种子。

下面，我们来看看交叉验证的实际用法。这里我们仍然借用第 7 讲中预测用户 LTV 的线性回归模型 model_lr，我们先用 KFold.split 对数据集进行拆分，然后对每一折循环，训练出不同的模型，最后给出一个验证分数。完整的代码你可以从这里下载。

复制代码

```
from sklearn.model_selection import KFold #导入K折工具
from sklearn.metrics import r2_score #导入R2分数评估工具
kf5 = KFold(n_splits=5, shuffle=False) #5折验证
i = 1
for train_index, test_index in kf5.split(df_LTV):
    X_train = df_LTV.iloc[train_index].drop(['年度LTV'],axis=1) #训练集X
    X_test = df_LTV.iloc[test_index].drop(['年度LTV'],axis=1) #验证集X
    y_train = df_LTV.iloc[train_index]['年度LTV'] #训练集y
    y_test = df_LTV.loc[test_index]['年度LTV'] #验证集y
    model_lr.fit(X_train, y_train) #训练模型
```

```
    print(f"第{i}折验证集R2分数：{r2_score(y_test, model_lr.predict(X_test))}")
    i += 1
```

输出如下：

复制代码

```
第1折验证集R2分数：0.5143622747243847
第2折验证集R2分数：-0.16778272779470416
第3折验证集R2分数：0.23879516275929713
第4折验证集R2分数：0.2482389409435588
第5折验证集R2分数：0.03088299924007265
```

在实际实战中，sklearn 还提供了更简单的方式，我们不用进行上面的 KFold 拆分过程，只用一个 cross_val_score 函数就能直接完成模型的 K 折拆分、训练和验证，一次性得到交叉验证结果。

接下来，我们再用这种方式来评估一下刚才的线性回归模型 model_lr：

复制代码

```
from sklearn.model_selection import cross_val_score # 导入交叉验证工具
# from sklearn.metrics import mean_squared_error #平均绝对误差
model_lr = LinearRegression() #线性回归模型
scores = cross_val_score(model_lr, #线性回归
                  X_train, #特征集
                  y_train, #标签集
                  cv=5, # 五折验证
                  scoring = 'neg_mean_absolute_error') #平均绝对误差
for i, score in enumerate(scores):
    print(f"第{i+1}折验证集平均绝对误差： {-score}")
```

这里我们把线性回归模型作为参数传入 cross_val_score 函数中，同时采用 5 折验证，计算出每一次验证的平均绝对误差，也就是预测值和真值之间的平均差异，误差绝对值越大，效果越不好。

输出的每一折平均绝对误差值如下：

复制代码

```
第1折验证集平均绝对误差：3191.99882739
```

```
2 第2折验证集平均绝对误差：1402.45679102
3 第3折验证集平均绝对误差：1168.49187113
4 第4折验证集平均绝对误差：1546.15537555
5 第5折验证集平均绝对误差：1138.41271054
```

看的出来，采用不同的训练集和验证集组合，得到的模型分数不同。最后，我们再把这 5 折的验证结果进行平均，得到 5 折交叉验证的平均分数。这个分数，将比仅拆分一次训练集 / 验证集得到的评估结果更为稳定。

我这里要强调的是，交叉验证虽然一直在用不同的数据拆分进行模型的拟合，但它实际上并不是在试图训练出新的模型，它只是我们对模型的一种评估方式而已。

好了，关于交叉验证，我们就说到这里，下面我讲一个对你来说非常实用的问题：如何调参。

网格搜索：直到找到最优的参数

一个复杂的机器学习模型，有可能有一大堆超参数，模型的性能很大程度上取决于超参数的值。什么超参数是最好的？那可不一定，因为针对不同的问题而言，最佳的参数一直在变化，而且基本上没有太多规律可循。那些机器学习老手很可能会告诉你：调参可是要靠感觉、靠直觉的，你调的多了，感觉也就出来了。

这样的答案会让咱们这种新手十分头疼，没有什么经验怎么针对当前机器学习任务和当前数据集找到最好的参数？难道要手工的一个一个尝试吗？我们之前学过的线性回归模型，只有两个主要超参数，而且是布尔型变量，组合起来是 4 种情况，这还可以调一调。但是像随机森林这种比较复杂的模型，参数非常多，怎么可能一个个去试呢？

你不用担心，**scikit-learn 中有一个 GridSearchCV 工具，中文叫做网格搜索，堪称是辅助我们自动调参的神器**，它可以帮我们自动调参，轻松找到模型的最优参数。

那么，怎么用这个网格搜索呢？我们需要做的有这几步：

第一步，明确你选择的模型有哪些可调的参数，这个你可以通过查看算法的说明文档来确定；

第二步，把模型中各超参数不同取值的排列组合尽可能多地列出来；

第三步，调用 GridSearchCV 工具，把可能的排列组合都传进去。

完成这三步后，GridSearchCV 会在后台创建出一大堆的并行进程，挨个执行各种超参数的组合，同时还会使用交叉验证的方法（名称中的 CV，意思就是 cross validation），来评估每个超参数组合的模型。最后，GridSearchCV 会帮你选定哪个组合是给定模型的最佳超参数值。

好，下面我们就基于第 7 讲中的 LTV 预测项目，看看怎么用 GridSearchCV 选出随机森林模型比较好的参数组合（在之前的实战中，模型的参数都选择的是默认值）。

在调用 GridSearchCV 调参之前，我们首先找出随机森林回归模型中有哪些参数，最好的方法还是去sklearn 官网查看。我们在这儿拿出一些重要的超参数来调一下：

参数	说明
n_estimators	int型，默认值=100，代表森林中的树木数量。
max_depth	int型，默认值=None，代表树的最大深度。
min_samples_split	int型 或 float型，默认值=2，代表分裂一个内部节点所需的最小样本数。
min_samples_leaf	int型 或 float型，默认值=1，代表叶节点所需的最小样本数。
max_features	可以是int型或float型，或者“auto”，“sqrt”，“log2”的任何一个值，用于设定寻找最佳分割时要考虑的特征数量。

我们知道，随机森林是由多棵决策树集合而成的算法，所以，这些参数都和各决策树的生成和剪枝过程相关。上面这些参数，我们并不一定要完全理解它们的作用，因为我们的重点不是去详细了解每一个超参数，而是让 GridSearchCV 自动找到对于我们当前任务合适的超参数组合。

找出随机森林模型的参数后，接下来，我们需要在程序中定义一个字典对象，列出各个超参数，以及我们希望去尝试的值组合，你可以参考下面的代码。在这段代码中，我定义了 rfr_param_grid 字典对象，并在字典中指定了一系列参数值的组合，举例来说，对于 n_estimators 这个参数我就选择了 100 和 300，当然你也可以尝试任意其它的值。

复制代码

```
model_rfr = RandomForestClassifier() # 随机森林模型
# 对随机森林算法进行参数优化
rf_param_grid = {"max_depth": [None],
                 "max_features": [3, 5, 12],
                 "min_samples_split": [2, 5, 10],
                 "min_samples_leaf": [3, 5, 10],
                 "bootstrap": [False],
                 "n_estimators" :[100,300],
                 "criterion": ["gini"]}
```

然后，在要调用 GridSearchCV 来进行超参数搜索之前，我们先看看 GridSearchCV 这个函数本身有什么需要设定的参数。

GridSearchCV 参数	含义
estimator	传递要检查超参数的模型实例。
params_grid	保存要尝试的超参数的字典对象。
scoring	要使用的评估指标，可以简单地传递有效的字符串/评估指标对象。
cv	交叉验证多少折。
verbose	设置为1，则表示在跑 GridSearchCV 时输出详细的信息，否则在进行调参时保持为静默状态。
n_jobs	跑 GridSearchCV 时并行运行的进程数，如果设定为-1，将使用所有可用的处理器（请谨慎使用-1）。

下面我们来调用 GridSearchCV 这个函数：

复制代码

```
from sklearn.model_selection import GridSearchCV # 导入网格搜索工具
model_rfr_gs = GridSearchCV(model_rfr,
                            param_grid = rfr_param_grid, cv=3,
                            scoring="r2", n_jobs= 10, verbose = 1)
model_rfr_gs.fit(X_train, y_train) # 用优化后的参数拟合训练数据集
```

在上述代码中，我将 GridSearchCV 返回的最佳参数组合存储在了 rfr_gs 这个新的随机森林模型。

然后，系统就会自动计算每种超参数组合拟合出的模型的准确率 / 损失：

复制代码

```
Fitting 3 folds for each of 432 candidates, totalling 1296 fits
[Parallel(n_jobs=10)]: Using backend LokyBackend with 10 concurrent workers.
[Parallel(n_jobs=10)]: Done  30 tasks      | elapsed:   22.0s
[Parallel(n_jobs=10)]: Done 180 tasks      | elapsed:  2.0min
[Parallel(n_jobs=10)]: Done 430 tasks      | elapsed:  4.8min
[Parallel(n_jobs=10)]: Done 780 tasks      | elapsed:  8.7min
[Parallel(n_jobs=10)]: Done 1230 tasks      | elapsed: 12.6min
[Parallel(n_jobs=10)]: Done 1296 out of 1296 | elapsed: 13.2min finished
```

你要注意的是，随机森林模型可能的超参数组合非常多，因此这个训练过程在我们的电脑上可能会持续很久很久，所以我特意没有设定太多的参数组合，而是选取了其中一部分。不过，这个训练过程也是花了十几分钟才搜索完。

经过 GridSearchCV 自动地换参、拟合并自动交叉验证评估后，最佳参数组合实际上已经被选出了，它就被存储在 rfr_gs 这个新的随机森林中，我们可以直接用它来做预测。这里，我们调用 rfr_gs 的 best_params_ 属性，来看一下这个最优模型是由哪些超参数组合而成的：

复制代码

```
print(" 最佳参数组合:", rf_gs.best_params_)
```

输出如下：

复制代码

```
最佳参数组合: {'bootstrap': True, 'max_depth': 10, 'max_features': 'sqrt', 'min_
```

然后，我们看看 GridSearchCV 模型的准确性。

复制代码

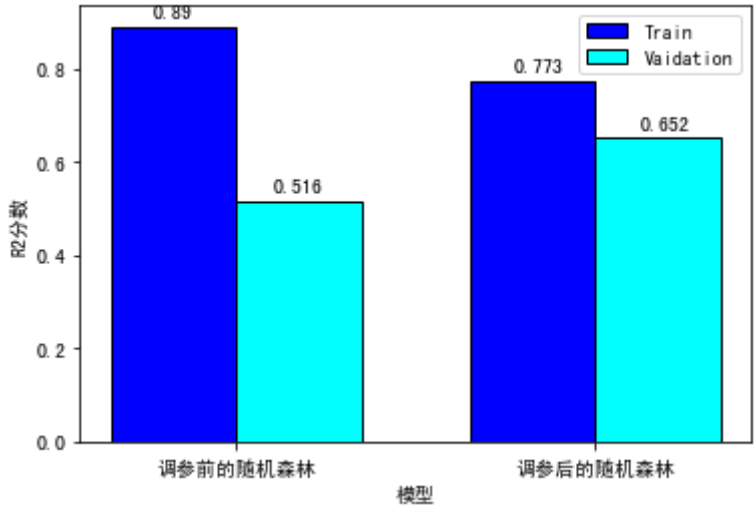
```
from sklearn.metrics import r2_score,  median_absolute_error #导入Sklearn评估模
print('训练集上的R平方分数-调参后的随机森林: %0.4f' % r2_score(y_train, model_rfr_gs
print('测试集上的R平方分数-调参后的随机森林: %0.4f' % r2_score(y_valid, model_rfr_gs
```

输出如下：

复制代码

```
训练集上的R平方分数-随机森林: 0.7729
测试集上的R平方分数-随机森林: 0.6523
```

这个分数需要和以前只调用随机森林模型默认参数的结果进行比较，所以，在这里我对这个模型训练了两次：一次不使用 GridsearchCV（使用默认超参数）；另一次我们使用 GridSearchCV 找到超参数最佳值。下面我们比较一下这两次的分数：



结果显示，经过网格搜索参数调优之后，虽然训练集上的分数降低了，但是在验证集上面的分数提升了，真棒！

学到这里，你可能会认为上面这一组参数是随机森林回归模型超参数的最佳值。但事实并非如此，上述超参数对于我们正在处理的数据集可能是最佳选择，但是对其他数据集未必适用。

到这里，我们就学完了两个优化技巧，再加上我们在前面两讲中学的特征工程和正则化技术，你在机器学习性能调优这块，就有了基本的理论和实践知识。恭喜你正式闯过“变现关”！

总结一下

这一讲中，我们学了两个和模型优化相关的方法，分别是交叉验证和网格搜索。

交叉验证是机器学习训练和验证模型时有效利用数据的方法，它的基本思路就是重复的使用数据，把样本数据进行切分，然后重组为不同的训练集和验证集。 在这个过程中，某次训练集中的某个样本在下次就可能成为验证集中的样本，所以被称作“交叉”。

网格搜索是帮我们优化模型超参数的技术，它在本质上是一种穷举法。对于每个超参数，我们都会指定一些可能的值。然后，GridSearchCV 会组合这些参数值得到若干组超参数，并使用每组超参数来训练模型，最后挑选出验证集上误差最小的超参数，来作为该模型的最优超参数。

这两个工具使用起来非常简单，你可以通过代码反复演练它们，尝试各种不同的验证折数和模型超参数。

那截止到这节课，我们实际上已经学习了四个提升机器学习模型效率的法宝，它们都特别重要。在这里我也帮你复习一下：

1. **特征工程**：目标是提高数据特征与模型的匹配度，我们总结了特征选择、特征变换和特征构建三个思路来帮助你解决具体问题。
2. **防过拟合**：过拟合就是模型针对训练集，拟合程度过高，失去了在新数据集上泛化的能力。我们的模型要在拟合和泛化之间保持一个平衡。
3. **交叉验证**：交叉验证可以帮我们利用好有限的数据集。你要注意，我们说的过拟合也有可能不是模型导致的，而可能是因为数据集划分不合理造成的，这在比较小的数据集上尤为明显。所以，交叉验证所得到的多次评估结果，能够避免某一次数据集划分不合理所带来的偏差。
4. **参数调优**：每一种算法都有属于自己的一系列可调超参数（外部参数），你要尽力找到对于当前数据集而言最好的一套参数。

在下一讲中，我们将开启激活关，让我们一起期待吧！

思考题

这节课就到这里了，最后，我想给你留三个思考题：

1. 除了最简单的交叉验证函数 cross_val_score 之外，sklearn 还提供另外一个函数 cross_validate，这个函数和 cross_val_score 的区别有两点：一是它允许指定多个评估指标；二是它返回的信息更丰富，除评估分数之外，它还返回一个包含拟合时间、分数时间、以及可选的训练集分数和拟合模型的字典。请你尝试使用这个函数，并分析它返回的信息。

提示：

```
    scores = cross_validate(lasso, X, y, cv=3,
                            scoring=('r2', 'neg_mean_squared_error'),
                           return_train_score=True)
```

2. 除了普通的 KFold 方法之外，还有重复 K 折 RepeatedKFold、留一法分折 LeaveOneOut、留多法分折 LeavePOut 等变体，也可以为数据分折，那么。请你尝试用这些方法来拆分数据，做交叉验证。

提示：

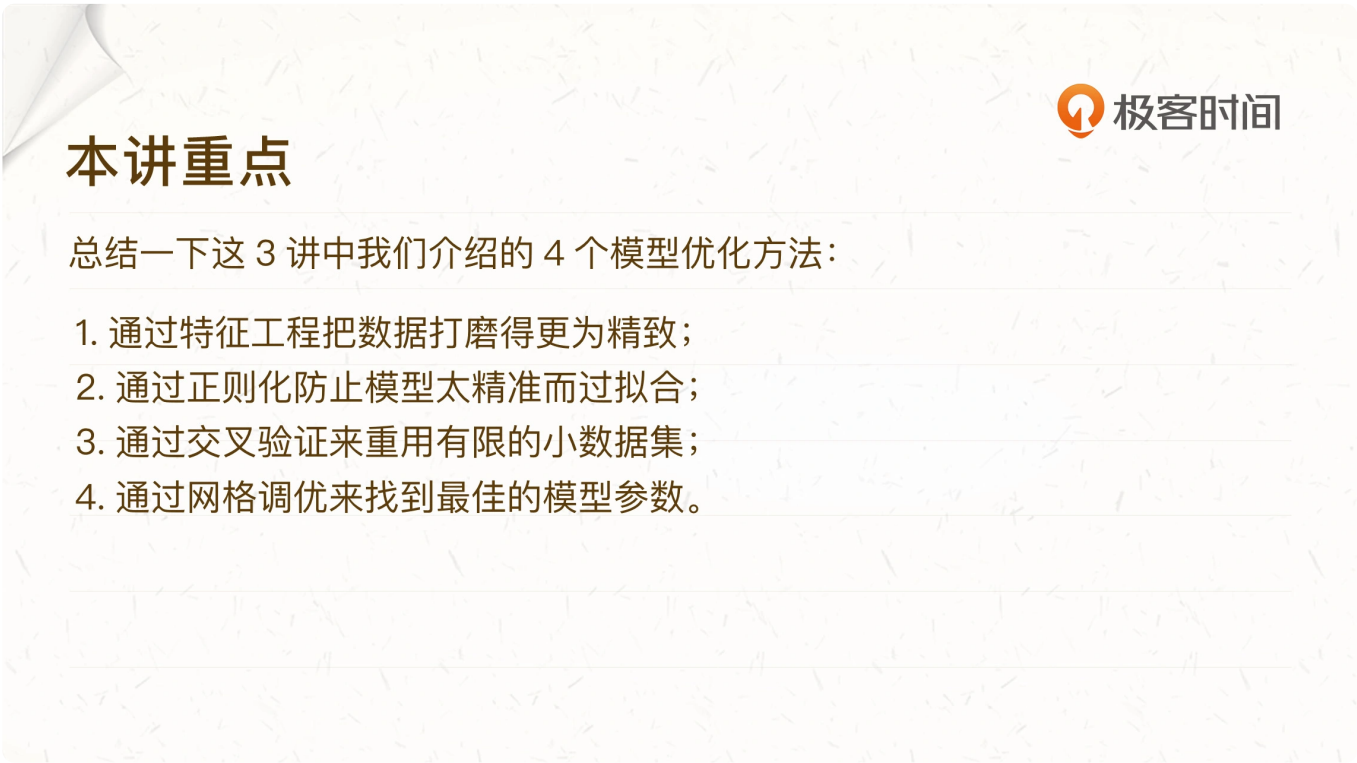
```
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import LeavePOut
...
>>> loo = LeaveOneOut()
>>> for train, test in loo.split(X):
...     print("%s %s" % (train, test))
```

3. 除 GridSearchCV 之外，sklearn 还提供了另一个网格搜索 API，叫 RandomizedSearchCV，中文名是随机搜索。对于完全相同的参数空间，随机搜索运行时间比起网格搜索会大大降低，这是因为它采用的是随机参数组合。不过，随机搜索返回的模型性能可能会稍差，这就是时间和最优性能的一种权衡。请你尝试使用 RandomizedSearchCV 进行搜索，找到比较好的超参数。

提示：

```
from sklearn.model_selection import RandomizedSearchCV
random_search = RandomizedSearchCV(clf, param_distributions=param_dist,
                                   n_iter=n_iter_search)
```

欢迎你在留言区分享你的想法和收获，我在留言区等你。如果这节课帮到了你，也欢迎你把这节课分享给自己的朋友。我们下一讲再见！



分享给需要的人，Ta订阅后你可得 20 元现金奖励

赞 0 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 09 | 模型优化（中）：防止过拟合，模型也不能太精细

精选留言

写留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。